PVT Variations Aware Clock Tree Synthesis in the Presence of Routing Obstacles

Manuscript Submitted to ICCAD2005

***Abstract*— This paper describes a clock tree synthesis methodology for high performance ASICs. The main goal is to produce process, voltage, and temperature (PVT) variations tolerant clock distribution network in the presence of complex rectilinear routing obstacles. We introduce three key ideas. First, we note that not all data paths between registers have same sensitivity to the clock uncertainty. [22] The proposed methodology respects this difference, and let those more sensitive to the clock uncertainties share longer common path in the clock tree. Second, we use extended Delaunay triangular mesh to represent the physical proximity of clock sinks in the presence of routing obstacles. Based on the physical proximity information and timing constraint information, we use recursive graph partitioning to generate the initial clock tree topology. Third, we adopt multi-level optimization techniques to refine the clock tree topology and physical embedding. The topology and embedding is optimized with accurate timing and wire length estimation. Experimental results show significant improvements on PVT variation tolerance with little wirelength overhead.**

I. Introduction

A. Motivations

With rapidly increasing clock frequency, the clock uncertainties introduced by process, voltage, and temperature (PVT) variations consume significant portion of a clock cycle time and consequently decrease the circuit performance [4]. Designing a PVT variation tolerant clock distribution network becomes a vital part of high performance digital circuits. Recently, non-tree clock distribution topologies [16] [18] have been proposed to reduce the clock uncertainty by adding shunt connections. Some of them have been applied to the high performance processor designs [15][19][1]. However, for the automated ASIC design flow, the tree structure is still favorable for following two reasons. First, the tree structure is easier to analyze and can be integrated into current static timing analysis flow. Second, tree consumes less routing area and hence has lower power consumption and causes less routability problems.

For clock tree synthesis and routing algorithms, one practical challenge is the existence of routing obstacles. The state-of-the-art SoCs usually consist of a number of memory blocks and IP macros. Some of these blocks form the routing obstacles for clock wires. If the clock topology generation algorithm does not take the routing obstacles into account, resulted clock tree may have a lot of undesired wiring detours, which may cause severe skew and routability problemes. In this paper we present an automated clock tree synthesis tool for high performance digital circuits. The tool handles rectilinear routing obstacles and produces PVT variation tolerant clock tree with short total wire length.

B. Previous Works

As a classic CAD problem, clock tree synthesis has received intensive research efforts in the past twenty years. The proposed clock tree topology generation algorithms can be roughly classified into three categories: top-down partitioning, bottom up merging, and iterative searching. The main objectives are balancing the load, minimizing the total wire length and delay.

The median and mean method (MMM) proposed by Jackson et *al.* [11] recursively partition the clock sinks on a plane according to the locations. This method produces a well balanced topology. However, it does not consider routing obstacles.

The geometric matching [12] and greedy-DME [6], uses bottom -up matching to construct the tree topology. These methods depend on the dynamic nearest neighbor queries, which is computationally expensive in the presence of routing obstacles. In a Manhattan plane without obstacles, the shortest path query only takes O(1) time, while with obstacles, the fastest algorithms have a time complexity of at least $O(m \log m)$, where m is the number of corner nodes of all obstacles.

Ellis et *al.*[7] and Chou et *al.* [5] both used simulated annealing to search for the optimal tree topology. Multiple objectives are optimized simultaneously in the simulated annealing framework.

Recently, several works target at the variations aware clock tree synthesis. Velenis et *al.* [21] first noticed that not all datapath have the same sensitivity to the clock uncertainty, and use a sequential merging scheme to construct the clock tree topology. Their method does not use any physical proximity information. It may results in excessively large total wire length, and the tree topology may be very unbalanced.

In [10], Hu et *al.* extended the DME [2] algorithm to accommodate the permissible clock uncertainty constraints for a given clock tree topology. This method can significantly reduce the timing violations caused by the

process variations on the interconnect. However, since the tree topology is generated by the non-variations-aware DME algorithm, it could not reduce the clock uncertainties caused by the voltage variations on the clock buffers, which is believed to be one main cause of clock uncertainties. In [3], the clock sinks are partitioned into groups, the algorithm reduces the wirelength by only minimizing intra-group skew values.

C. Our Contributions

As stated in [21], only small portions of the datapaths in synchronous circuits are most sensitive to the clock uncertainties. For those registers, we ought to put them topologically close to each other in the clock tree. At the same time, in order to minimize the total wire length, we also need to consider the physical proximity between clock sinks when we construct the clock tree. We extend the basic Delaunay triangulization by adding virtual nodes to the boundary of the obstacles. The extended Delaunay triangular mesh provides a graph representation of the spatial relation of the clock sinks. We combine the Delaunay triangular mesh with the clock uncertainty constraint graph and use graph partitioning to balance the PVT variation tolerance and total wire length cost. We use multi-level optimization to further refine the clock tree topology with actual physical embedding information. We utilize the accurate wirelength and delay estimations based on actual physical embedding to guide the search of optimal topology. The main contributions of this work are:

- We explicitly address the requirement of both clock uncertainties and wirelength minimization in the clock tree topology generation. By considering both the spatial and temporal relations between clock sinks, we reduce the number of timing violations by 88% with only 1.5% of wire length increasing.
- We use extended Delaunay triangular mesh to represent the physical proximity information in the presence of routing obstacles.
- We adopt a multi-level optimization technique to simultaneously refine the clock tree topology and embedding. With multi-level optimization technique, we are able to synthesis clock tree with 200K flip-flops, 1.6 million timing constraints, and 200 routing obstacles within 6 hours.

The rest of this paper is organized as follows. In Section II, we formulate the PVT variations aware clock tree synthesis problem. We then describe the flow of the methodology in Section III. In Section IV, we present the initial clock tree topology generation by recursive graph partitioning. After that, we introduce the multilevel optimization framework for clock tree topology refinement and physical embedding. We show experimental results in Section VI. Finally, we conclude the paper in Section VII.

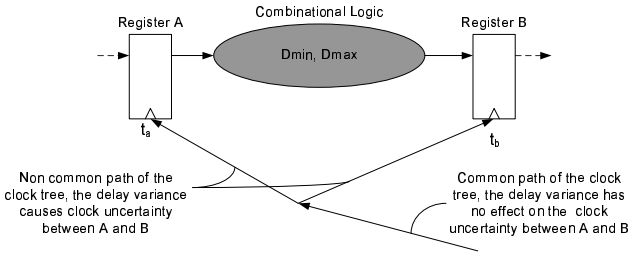

Fig. 1. Clock Uncertainties and Timing Constraints in Sequential Circuits

II. Problem Statement

We formulate the PVT variation aware clock tree synthesis problem in this section. In subsection A, we clarify the definitions of skew and clock uncertainty, and discuss their effect on circuit behavior. In subsection B, we present the problem formulation.

A. Clock Uncertainties and CRPR

We make distinction between clock skew and clock uncertainties. In this paper, we refer clock delay to the signal propagation delay from the clock source to the clock input pins. For a pair of registers, A and B, the difference of clock delay t_a and t_b can be decomposed into two parts, the deterministic part and the probabilistic part. We call the deterministic part $t_a - t_b$ the skew, which is due to the designed mismatch of delay and can be calculated using nominal design values. We call the probabilistic part, $\Delta_{a,b}$, the clock uncertainty, which is caused by the PVT variations. Figure 1 shows a schematic of a datapath in sequential circuits. Assuming the combinational logic has maximal delay Dmax and minimal delay Dmin, we have following two timing constraints.

$$t_a + D_{max} + t_{setup} - t_b + \Delta_{a,b} \leq t_p \quad (1)$$

$$t_a + D_{min} - t_b - \Delta_{a,b} \geq t_{hold} \quad (2)$$

According to above two constraints, the clock skew $t_a - t_b$, may be useful and can be introduced intentionally [8]. On the other hand, the clock uncertainties $\Delta_{a,b}$ are always harmful to the performance and reliability of the circuits. Traditionally, the designers use a certain portion of the clock delay, e.g. 15% of the insertion delay, as a safe margin of the clock uncertainty estimation. However, this approach overestimates the clock uncertainty. From Figure II-A, we see that the delay variations on the common path in the clock tree do not contribute to the clock uncertainty between two registers. Only the variations on the distinct paths make the contribution. The state-of-art static timing analyzer[22] has

already implemented the clock reconvergence common path removal (CRPR) algorithm, which can exclude the pessimism introduced by considering the variations on the common path. In this paper, we use a constant portion of the delay from the nearest common ancestor to the clock sink as the estimation of the clock uncertainty between two registers. We call this constant as the clock uncertainty coefficient, k.

B. PVT Variation Aware Clock Tree Synthesis Problem

Let $S = \{s_i = (x_i, y_i)\}$ denote a set of clock sinks on a Manhattan plane and s_0 the clock source. Each clock sink s_i has a capacitive load C_i. The routing blockages B are a set of rectangles on the plane. And $u_{i,j}$ is the maximum permissible clock uncertainty between register i and register j. Clock tree $T(S)$ is a tree rooted at s_0 and spanning on the union set of Steiner points S' and clock sinks S. We formulate the PVT variations aware clock tree synthesis problem as follows.

PVT Variations Aware Clock Tree Synthesis Problem: *Given a set of clock sinks S=(xi,yi,) on a Manhattan plane, a set of rectangular routing blockage B, the permissible clock uncertainty between clock sinks $u_{i,j}$, the clock uncertainty coefficient k, construct a buffered clock tree T(S), such that the total wirelength is minimized while all the permissible clock uncertainty constraints are met.*

An alternative way to formulate this problem is to maximize minimum slackness for given total wire length or power budget. This formulation is a dual of our formulation, and the proposed method can solve both problems.

III. OVERALL FLOW OF THE METHODOLOGY

Figure 3 shows the pseudo code of our PVT variations aware clock tree synthesis algorithm. The algorithm inputs the clock sink distributions, routing obstacles, and maximal pairwise clock uncertainty constraints. it proceeds mainly in three steps. First, it extracts the spatial relation and temporal relation between clock sinks using extended Delaunay triangulization, and generate the initial tree topology by recursive partitioning. The tradeoff between wire length and PVT variation tolerance is controlled by the net weighting. Then, a multi-level optimization scheme refines the tree topology and routing with accurate physical embedding information. Finally, we tune the buffer size and wire size to further improve the solution quality.

IV. INITIAL TREE TOPOLOGY GENERATION

In this section, we describe our method for initial clock tree topology generation. In subsection A, we first introduce the extended Delaunay triangulization and prove several useful properties of it. Then, we describe our initial topology generation algorithm in subsection B.

Algorithm: Variation Aware Clock Tree Synthesis
Input: Clock sinks set S
Routing obstacles set B
Clock uncertainty constraint graph Gc.
Output: clock tree T
Procedure:
1. Generate extended Delaunay triangulization Gs = (S, Es), and assign edge weights
2. Estimate the minimal size of the bottom level cluster, s_min
3. Construct the uncertainty constraint graph Gt = (S, Et), and assign edge weights
4. while (cluster size > s_min) do recursive partitioning on G = (S, Et + Eu)
5. While not converged
{
5.a For (i = 1 to n) do
Low temperature simulated annealing to optimize the permutation of level *i* subtrees;
5.b For (i = n to 1) do
Low temperature simulated annealing to optimize the permutation of level *i* subtrees
}
6. Clock tuning

Fig. 2. PVT Variations Aware Clock Tree Synthesis Algorithm

A. Extended Delaunay Triangulization

Delaunay triangulization and its dual format, Voronoi diagram, have been used in several clock tree synthesis algorithms for simple representation of near neighbor information of objects in a 2D space [7]. However, in large ASIC designs, the existence of various routing blockages distorted the original distance matrix and makes the distance computation much more expensive.

The Voronoi diagram problem in the presence of obstacles is referred as geodesic Voronoi diagram problem. Many algorithms have been proposed for the construction. However, most of them require the support of complicated data structures [9]. In our application, we only need a basic representation of the geometrical proximity structure. We extended the basic Delaunay triangulation by adding virtual nodes to the boundaries of the obstacles.

In order to describe the virtual nodes adding scheme, we first introduce the concept of a *shadow region*. Figure IV-A shows an example of a *shadow region*. Without loss of generality, we assume that AB is the right boundary of a rectangular obstacle. We draw a square $AA'B'B$. Let

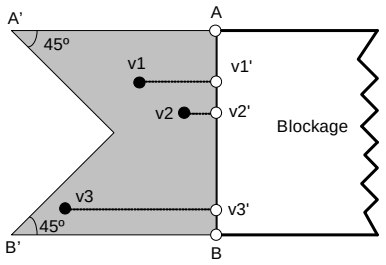

Fig. 3. *Shadow region* and direct virtual node insertion

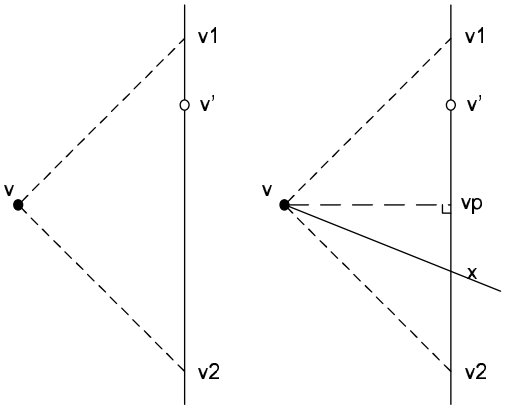

Fig. 4. Proof of Theorem 1

O be the center of the square. We take away the triangle $OA'B'$, the rest of the square is called the *shadow region* of obstacle boundary *AB*.

Given a set of points, V, and a set of rectangular obstacles, B, we construct the extended Delaunay triangular mesh in following steps. First, we add every corner point of the obstacles to the virtual nodes set V'. Then, we construct the shadow region for each obstacle boundary segment. For every node v_i lies inside the shadow region of obstacle boundary *AB*, we obtain its projection, v_i', on *AB* and add virtual node v_i' to the virtual node set V'. We compute the Delaunay triangulization on point set $V+V'$. By deleting all of the edges crossing obstacles, we obtain the extended Delaunay triangular mesh $G_s = (V+V', E_s)$. For the correctness of our virtual node insertion and edge deletion scheme, we prove following lemmas and theorems.

Lemma 1: *Assume an obstacle boundary edge, l, and a node v. From node v, draw two lines each with the slope of $+1$ and -1. Let these two lines intersect l at $v1$ and $v2$, respectively. If there is a virtual node v' between $v1$ and $v2$, then there is not edge in the Delaunay triangular mesh incident to v and intersects l.*

Sketch of the proof: From Figure 4, we see that if there is an edge of Delaunay triangular mesh starting from v intersect with l at point x. The Manhattan distance between v' and x is always smaller than the Manhattan distance between v and x, which contradicts to the definition of Delaunay triangulization. Hence, such edge does not exist.

From Lemma 1, we prove following theorems. The first theorem shows the correctness of virtual node insertion, and the second theorem shows that the graph will not become disconnected after edge deletion.

Theorem 1: *Assume an obstacle boundary edge, l, and a node v. From node v, draw two lines each with the slope of $+1$ and -1. Let these two lines intersect l at $v1$ and $v2$, respectively. If there is a virtual node v' between $v1$ and $v2$, then there is not edge in the Delaunay triangular mesh incident to v and intersects l.*

Theorem 2: *The extended Delaunay triangular mesh is connected after deleting all the obstacle crossing edges.*

Assume there are n points and k rectangular obstacles on the plane. The extended Delaunay triangulization has following properties.

Lemma 2: *The total number of virtual nodes is $O(n)+O(k)$*

Lemma 3: *The extended Delaunay triangulization can be conducted in $O((n+k)\log(n+k))$ time.*

Lemma 4: *On an extended Delaunay triangular mesh $G = \{V+V', E\}$, if shortest paths between node A and node B are all* non-x-monotone *or* non-y-monotone, *all these paths must contain at least one virtual node.*

B. Topology Generation through Graph Partitioning

We take the union of the extended Delaunay triangular mesh and clock uncertainty constraint graph, and use recursive graph partitioning to generate the tree topology. The key of this process is the edge weight assignment. For a Delaunay triangular mesh $G = (V+V', E)$ we assign the edge weight using following equation:

$$c(e) = \frac{MAXdist(e')}{dist(e)^{\alpha}} \tag{3}$$

The intuition behind this equation is that the cost of separating two nodes should be inversely proportional to the physical distance between them. In our implementation, we set the value of α to 1.0. For clock uncertainty constraint graph $G = \{V, E\}$, the edge weight is set using equation $c(u,v) = ke^{\frac{1}{\Delta u,v}}$, where, $\Delta u,v$ is the maximal permissible clock uncertainty between node u and node v. The constant k is decided by the ratio of the total edge weight of G_t and G_s. In our implementation, the value of k is set to let the total edge weight of G_t and G_s to be equal. The k value decides the tradeoff between PVT variations tolerance and wire length. Choosing the larger k implies higher preference for variations tolerance while the smaller k gives the preference to shorter wire length.

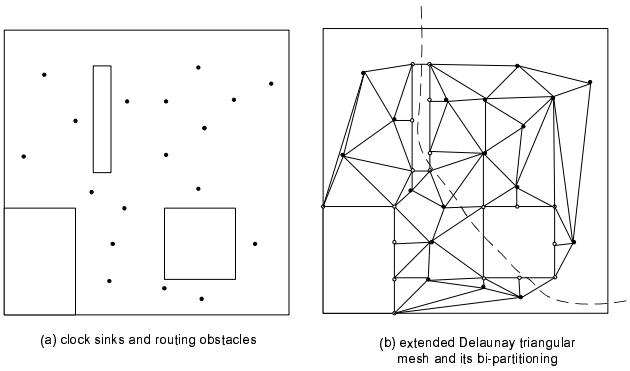

Fig. 5. Extended Delaunay Triangulization and a Bi-Partitioning of Triangular Graph

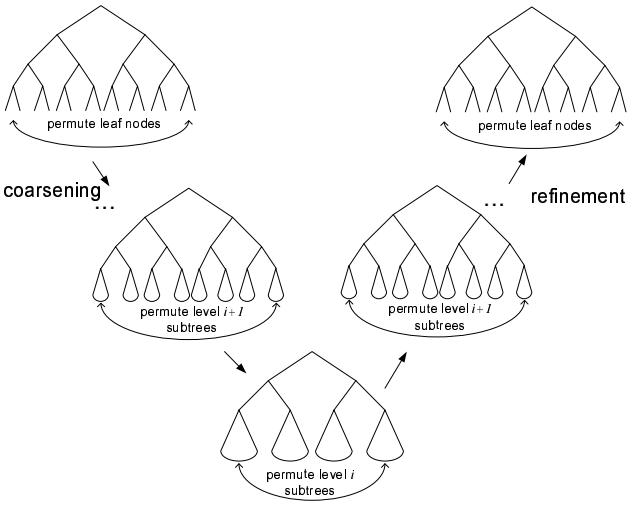

Fig. 6. A v-cycle of multilevel clock tree refinement

Note that when constructing the Delaunay triangular mesh, we do not distinguish the flip flops and the virtual nodes. The weight of each edge is assigned purely based on the physical distance. The flip flops and the virtual nodes are distinguished by different node weight. When we do partitioning, the node weight of a flip flop is proportional to its capacitive load, while all the virtual nodes have zero weight.

Figure 5 shows an example of a bipartitioning on an extended Delaunay triangular mesh. Figure 5(a) illustrates the clock sinks distribution and routing obstacles. Figure 5(b) is the extended Delaunay triangular mesh and its 2-way partitioning. The dashed line illustrates the cut line of the graph. With this partition, the clock sinks are clearly partitioned into two parts by its physical location and no obstacle is immersed inside the points set of the same cluster.

V. MULTI-LEVEL REFINEMENT AND TREE EMBEDDING

The recursive graph partitioning presented in the previous section produces a good balance between physical proximity and temporary constraints. However, at the abstract level of graph partitioning, there is a lack of detail physical embedding information; the optimization can not be driven by accurate wirelength and delay cost. Hence, the topology generated is suboptimal. We adopt a multi-level optimization scheme to further improve the clock tree topology and embedding. While exploring the search space of different topologies, we simultaneously construct the physical embedding and insert buffers. The accurate wirelength and delay estimations guide the topology optimization. According to our experiments, the multi-level refinement can improve the total wirelength by 10% to 15% as well as reduce the number of timing violations by 10% to 30%.

Figure 6 illustrates the process of a ”v-cycle” in the multilevel clock tree refinement. The procedure starts from the finest level optimization, where the permutation of the leaf level nodes is optimized. Through a coarsening process, we optimize the tree topology at a higher and higher level. At level i, the permutation of the i^{th} level subtrees are optimized. When we finish the bottom level optimization, we go back to a refinement process, where the optimizations are performed at finer and finer level. We repeat this V-cycle several times, until we find a satisfied solution or the quality of solution can not be improved. Typically, it takes 4 to 6 V-cycles to optimize a clock tree with about 200K clock sinks.

At level i, a low temperature simulated annealing procedure optimizes the permutation of the i^{th} level subtrees. The cost function used for simulated annealing consists of three parts, the total wirelength, the insertion delay, and the clock uncertainty constraints violations. The fundamental movement of the simulated annealing is an exchange of subtrees rooted at the same level [5]. A fast clock tree embedding subroutine incrementally constructs the embedding of the tree and calculates the cost for simulated annealing.

In order to accelerate the wire length and delay estimation, we adopt a simplified tree embedding and buffer insertion scheme. The buffer insertion scheme is a simple fanout rule based insertion. We restricted the clock buffers to be placed only at the Steiner points, and enforce the load to input capacitance ratio for every clock buffer as a constant, for example, 4. This scheme requires $O(\log n)$ time for an incremental adjustment.

The embedding algorithm is based on bottom-up merging. At each non-leaf node in the tree structure, we record the location of the center of mass for the subtree rooted at that node. When we merge two nodes, we use Dijkstra’s shortest path algorithm on the Delaunay

graph to find a path between two nodes and let all L shaped connections bend toward the center of mass of the subtree rooted at its parent node. We use Tsay's zero skew emerging scheme [20] to decide the location of tapping points and apply wire snaking to balance the delay when needed. This scheme requires only $O(\log n)$ time for an incremental tree construction.

After obtaining the optimized topology, we can use DME algorithm to find the best embedding and use more accurate delay calculation methods to adjust the tapping point position and buffer sizes.

VI. Experimental Results

We implement the clock tree synthesis flow in C programming language. We use Metis library [14] for graph partitioning and Triangle 1.5 package [23] for Delaunay triangulization. The platform is a 2.4GHz Pentium 4 desktop running Linux.

We perform two sets of experiments. First, we compare the performance of our methodology with DME algorithm on a set of publicly available benchmarks [24]. This set of benchmarks does not contain routing obstacles. Then, we demonstrate the experimental results on a large synthetic benchmark. The design has 200K flip flops, 1.6million timing constraints and 200 rectangular routing blockages. The chip size is 14 mm by 14 mm. The capacitive load of each flip flop is 3.4fF.

We randomly generate the permissible clock uncertainty constraints for all pairs of flip flops. For any pair of flip flops, we assume there is a data path between them with probability 0.1. If there is a data path between two flip-flops, we assume the maximal permissible clock uncertainty between them is a random number uniformly distributed in the range of 1 to 1000 ps.

We generate clock trees using both our program and the DME algorithm. The original DME algorithm generates unbuffered clock trees. We use the same buffer insertion/sizing routines we used in our PVAT algorithm to insert the buffer on the clock trees generated by DME algorithm. We extract the SPICE netlist from the routed clock tree. We set all the R, C values, supply voltages, and transistors lengths to be random variables with Gaussian distribution. The 3σ values are 10% of their nominal values. To model the temperature variations, we sweep the environment temperature from 20C to 80C. Due to the simulation tool limitation, we did not model the effect of the on-chip temperature gradient. Part of its effect is reflected by the interconnect resistance variations. We perform Monte Carlo analysis using HSpice. We get the clock uncertainty by taking the maximal difference between clock delays of two flip-flops in 40 runs of HSpice simulations.

Table 1 are the comparisons between our method and DME algorithm. Column 2 to column 5 are the results

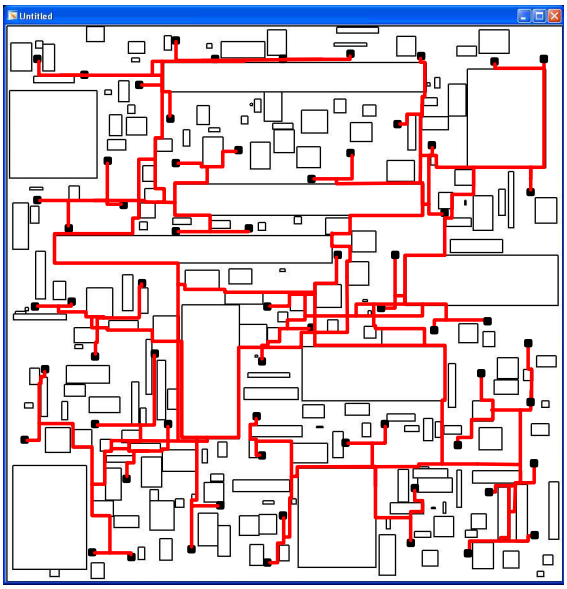

Fig. 7. Clock tree for a 200K flip-flops, 200 obstacles chip top level route are shown, small black squares indicate the root of local routing trees

of our algorithm. Column 6 to column 9 list the results for DME. The second and the sixth columns are the numbers of clock uncertainty violations. Our method on average has 12.6 violations for every circuit while the DME has 108. The third and the seventh column show the maximal violations. Our method reduces the average value of maximal violations by 70%. The fourth and eighth columns are total wirelengths. Comparing with DME, our method only increase the wirelength by 1.5%.

Figure 7 shows the top level routing with routing blockages for the large scale testcase. We can see from the figure that the existence of obstacles significantly distorted the physical proximity structure. Our program completes in 6 hours. Among the 1.6 million clock uncertainty constraints, only 65 of them are violated.

VII. Conclusions and Future Directions

We present a novel clock tree synthesis methodology for high performance ASICs. The proposed scheme recognizes different requirements on clock uncertainty by different data path and let the registers on the two ends of a critical path share more common path on a clock tree, thus decrease the effect of clock uncertainty caused by PVT variations. We use an extended Delaunay triangular mesh to represent the clock sink proximity in the presence of routing obstacles. Taking both spatial and temporal relations into consideration, we use graph partitioning to get an initial clock tree topology. We then refine the clock topology and embedding through a multi-level optimization process.

Experimental results show that our method produces

TABLE I

COMPARISONS BETWEEN OUR METHOD AND DME

ckt	PVAT				DME			
	#viol	Max Vio (ps)	w.l. (mm)	CPU (s)	#viol	Max Vio (ps)	w.l. (mm)	CPU (s)
r1	7	61	188.1	9.2	45	165	187.2	0.1
r2	5	107	362.3	16.0	63	331	355.0	0.3
r3	9	375	447.6	47.6	82	979	443.9	0.4
r4	14	214	906.1	103.4	295	642	894.2	2.6
r5	28	213	1340.9	189.7	457	1204	1316.9	7.4

only one ninth of the clock uncertainty violations comparing with DME algorithm. It reduces the maximal violations by 70% with only 1.5% increase in wirelength. We also demonstrate that our algorithm can complete a complex design with 200K flip-flops, 1.6 million timing constraints, and 200 rectangular routing obstacles in 6 hours of CPU time.

Our ongoing efforts include following:

- Integrate the proposed clock tree topology generation method with variation aware clock embedding/routing [7].
- Better post processing method including wire sizing, variations aware buffer insertion.
- Clock tree topology generation considering more grouping constraints, e.g. clock gating, multiple clock domains, etc.

REFERENCES

[1] D. W. Bailey and B. J. Benscheneider, *Clocking Design and Analysis for a 600-MHz Alpha Microprocessor,* IEEE Journalof Solid-State Circuits, Vol. 33, No. 11, pp.1627-1633, November, 1998

[2] K. D. Boese and A. B. Kahng, *Zero-skew Clock Routing with Wirelength Minimization,* in Proc. IEEE Int'l Conf. on ASICs, pp. 17-21, September, 1992

[3] Y. Chen, A. B. Kahng, G. Qu, and A. Zelikovsky, *The Associative-Skew Clock Routing Problem,* in Proc. ICCAD, pp. 168-172, November, 1999

[4] D. Chinnery and K. Keutzer, *Closing the Gap Between ASIC & Custom: Tools and Techniques for High-Performance ASIC Design*, Kluwer Academic Publishers, 2002

[5] N.C. Chou, and C. K. Cheng, "Wire length and delay minimization in general clock net routings," IEEE trans. On CAD, 1996

[6] M. Edahiro, *A Clustering-Based Optimization Algorithm in Zero-Skew Routings,* in Proc. DAC, pp. 612-616, June, 1993

[7] G. Ellis, L. T. Pilleggi, and R. A. Rutenbar, *A Hierarchical Decomposition Methodology for Multistage Clock Circuits*, in Proc. ICCAD, pp. 266-273, November, 1997

[8] J. P. Fishburn, *Clock Skew Optimization,* IEEE Trans. Computers, pp. 945-951, July 1990

[9] S. Guha and I. Suzuki, *Proximity Problems For Points On A Rectilinear Plane With Rectilinear Obstacles,* Algorithmica,vol. 17, pp. 281 - 307, 1997

[10] B. Lu, J. Hu, G. Ellis, and H. Su, *Process Variation Aware Clock Tree Routing,* in Proc. ISPD, pp. April. 2003

[11] M. A. B. Jackson, A. Srinivasan, and E. S. Kuh, *Clock Routing for High-Performance ICs*, in Proc. DAC, pp. 573-579, 1990

[12] A. B. Kahng, J. Cong, and G. Robins, *High Performance Clock Routing Based on Recursive Geometric Matching,* in Proc. DAC, pp. 322-327, 1991

[13] A. B. Kahng and C. W. Tsao, *More Practical Bounded-Skew Clock Routing,* in Proc. DAC, pp. 594-599, June 1997

[14] G. Karypis and V. Kumar, *Multilevel Algorithms for Multi-Constraint Graph Partitioning,* Technical Report TR 98-019, Department of Computer Science, University of Minesota, 1998

[15] N. A. Kurd, et *al.*, *A Multigigahertz Clocking Scheme for the Pentium 4 Microprocessor,* IEEE Journal of Solid-State Circuits, vol. 36, No. 11, pp. 1647 - 53, November 2001

[16] M. Mori, H. Chen, B. Yao, and C. K. Cheng, *A Multiple Level Network Approach for Clock Skew Minimization with Process Variations,* in Proc. ASPDAC, 2005

[17] A. Okabe, B. Boots, and K. Sugihara, *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams,* Wiley & Sons, 1992

[18] A. Rajaram, J. Hu, and R. Mahapatra, *Reducing Clock Skew Variability Via Cross Links,* in Proc. DAC, pp. 18 -23, 2004

[19] P.J. Restle, et *al. , The Clock Distribution of the Power4 Microprocessor,* ISSCC 2002, Session 8.4

[20] R. S. Tsay, *An Exact Zero-Skew Clock Routing Algorithm,* IEEE Trans. On CAD, Vol. 12, No. 2, pp. 242-249, February, 1993

[21] D. Velenis, E.G. Friedman, and M. C. Papaefthymiou, *A Clock Tree Topology Extraction Algorithm For Improving the Tolerance of Clock Distribution Networks to Delay Uncertainty,* in Proc. ISCAS, pp. 4.422-4.425, May, 2001

[22] J. Zejda and P. Frain, *General Framework for Removal of Clock Network Pessimism,* in Proc. ICCAD, 2002

[23] http://www-2.cs.cmu.edu/ quake/triangle.html

[24] http://vlsicad.ucsd.edu/GSRC/bookshelf/Slots/BST/